(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（Ａ）　(11)特許出願公開番号

特開平7－295871

(43)公開日　平成７年(1995)11月10日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　12/00	５１８　Ａ	7608－5B		
	５３３　Ｊ	7608－5B		
17/30				
		9194－5L	Ｇ０６Ｆ　15/ 40　３１０　Ｃ	

審査請求　未請求　請求項の数11　ＯＬ　（全 21 頁）

(21)出願番号　特願平7－23143

(22)出願日　平成７年(1995)２月10日

(31)優先権主張番号　２２８３１９

(32)優先日　1994年４月15日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　アラン・シー・アザギュリ
イスラエル20306ネシャ、ハシクマ・ストリート　32／6

(74)代理人　弁理士　合田　潔　（外２名）

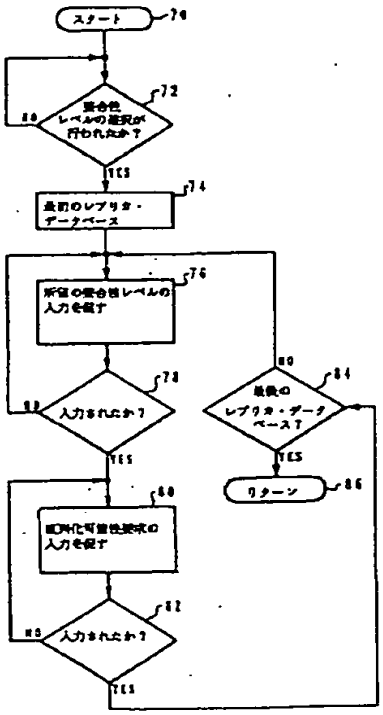
最終頁に続く

(54)【発明の名称】　データベース・アクセス効率の向上方法及びシステム

(57)【要約】

【目的】　分散データ処理システム内でのデータベース・アクセス効率を向上させる方法とシステムを提供する。

【構成】　分散データ処理システム内の１つまたは複数のロケーションで実施される回復可能なデータベース・システムにおいて、プライマリ・データベースとレプリカ・データベースとの間に維持される複数の多様な整合性レベルが指定される。ユーザは各レプリカに適した整合性レベルを選択できる。プライマリ・データベース内のレコードに対するアップデートがあると、各レプリカ内の対応するレコードに対するアップデートは、各レプリカに対して選択された整合性レベルに一致する方法で行われる。これにより、プライマリ・データベースと完全に整合性のあるレプリカが提供できる。あるいは、完全に整合性のあるデータを必要としないアプリケーションには、第２のレプリカを提供し、アクセス効率を向上させることができる。



1

【特許請求の範囲】

【請求項１】　分散データ処理システム内でのデータベース・アクセス効率を向上させる方法であって、
前記分散データ処理システム内において、ストアされた複数のレコードを持つ１つのデータベースを選択しプライマリ・データベースとして指定するステップと、
前記分散データ処理システム内の第２の物理的ロケーションにおいて前記プライマリ・データベースを複製するステップと、
前記プライマリ・データベースと前記の複製されたレプリカ・データベースとの間に維持される複数の多様な整合性レベルを指定するステップと、
ユーザに前記の複数の多様な整合性レベルから１つを指定させるステップと、
を有し、
前記プライマリ・データベース内のレコードに対するアップデートに応答して、前記プライマリ・データベースと前記レプリカ・データベースとの間に前記の複数の多様な整合性レベルの前記の指定された１つを自動的に維持する前記方法。

【請求項２】　前記分散データ処理システム内の第３の物理的ロケーションにおいて前記プライマリ・データベースを複製するステップをさらに有する、請求項１に記載の方法。

【請求項３】　ユーザに前記の複数の多様な整合性レベルから１つを指定させる前記ステップが、前記プライマリ・データベースとそのそれぞれのレプリカとの間に維持される前記の複数の多様な整合性レベルから別の１つを指定させるステップを有する、請求項２に記載の方法。

【請求項４】　分散データ処理システム内でのデータベース・アクセス効率を向上させるシステムであって、
前記分散データ処理システム内において、ストアされた複数のレコードを持つプライマリ・データベースと、
前記分散データ処理システム内の第２の物理的ロケーションにおいてストアされている前記プライマリ・データベースのレプリカと、
前記プライマリ・データベースと前記レプリカ・データベースとの間に維持される複数の多様な整合性レベルを指定する手段と、
ユーザに前記の複数の多様な整合性レベルから１つを指定させる手段と、
前記プライマリ・データベース内のレコードに対するアップデートに応答して、前記プライマリ・データベースと前記レプリカ・データベースとの間に前記の複数の多様な整合性レベルの前記の指定された１つを自動的に維持する手段と、
を有するシステム。

【請求項５】　前記分散データ処理システム内の第３の物理的ロケーションにおいてストアされている前記プライマリ・データベースの第２のレプリカをさらに有する、請求項４に記載のシステム。

2

【請求項６】　ユーザに前記の複数の多様な整合性レベルから１つを指定させる前記手段が、前記プライマリ・データベースとそのそれぞれのレプリカとの間に維持される前記の複数の多様な整合性レベルから別の１つを指定させる手段を有する、請求項５に記載のシステム。

【請求項７】　分散データ処理システム内でのデータベース・アクセス効率を向上させる方法であって、
前記分散データ処理システム内において、ストアされた複数のレコードを持つ１つのデータベースを選択しプライマリ・データベースとして指定するステップと、
前記分散データ処理システム内の第２の物理的ロケーションにおいて前記プライマリ・データベースを複製するステップと、
前記プライマリ・データベース内の１つの選択されたレコードに対するアップデートに応答して、前記の複製されたデータベース内の対応する選択されたレコードへのアクセスを自動的にロックするステップと、
前記の複製されたデータベース内の前記の選択されたレコードをアップデートするステップと、
前記アップデートの後で、前記の選択されたレコードへのアクセスをリリースするステップと、
前記分散データ処理システム内のユーザに、前記プライマリ・データベースおよび前記の複製されたデータベースの中のレコードを照会させるステップと、
を有し、データの絶対的な整合性を維持するとともにデータアクセスの使用可能性を向上させる方法。

【請求項８】　前記プライマリ・データベース内のレコードに対する各アップデートが、コミット・オペレーションを行う前の１つの整合性時点に戻り、前記の選択されたレコードへのアクセスを前記アップデートの後でリリースする前記ステップが、前記アップデートに続くコミット・オペレーションの後でのみ前記の選択されたレコードへのアクセスをリリースするステップを有する、請求項７に記載の方法。

【請求項９】　分散データ処理システム内でのデータベース・アクセス効率を向上させるシステムであって、
前記分散データ処理システム内において、ストアされた複数のレコードを持つプライマリ・データベースと、
前記分散データ処理システム内の第２の物理的ロケーションにおいてストアされている前記プライマリ・データベースのレプリカと、
前記プライマリ・データベース内の１つの選択されたレコードに対するアップデートに応答して、前記の複製されたデータベース内の対応する選択されたレコードへのアクセスを自動的にロックする手段と、
前記の複製されたデータベース内の前記の選択されたレコードをアップデートする手段と、
前記アップデートの後で、前記の選択されたレコードへ

のアクセスをリリースする手段と、
前記分散データ処理システム内のユーザに、前記プライマリ・データベースおよび前記の複製されたデータベースの中のレコードを照会させる手段と、
を有し、データの絶対的な整合性を維持するとともにデータアクセスの使用可能性を向上させるシステム。
【請求項１０】前記プライマリ・データベースを、いかなる時点でも、コミット・オペレーションを行う前の整合性時点に戻す手段をさらに有する、請求項９に記載のシステム。
【請求項１１】前記アップデートの後で前記の選択されたレコードへのアクセスをリリースする前記手段が、前記アップデートに続くコミット・オペレーションの後でのみ前記の選択されたレコードへのアクセスをリリースする手段を有する、請求項１０に記載のシステム。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、改善されたデータベース・アクセス制御の方法に関し、具体的には、プライマリ・データベースと１つまたは複数のレプリカ・データベースとの間の整合性を保つ改善された方法とシステムに関する。より具体的には、本発明は、回復力のある（resilient）データベース・システムにおいて、ユーザが、複数のレプリカ・データベースに付ける多様な整合性レベルを選択出来る、改善された方法とシステムに関する。
【０００２】
【従来の技術】多目的のマルチプロセッシング・コンピューティング・システムは、典型的には、通信網によって相互接続された複数のノードを含む。そのようなシステムでは、各ノードは、データ処理装置、データ記憶装置、および、複数の通信ポートを含むことができる。データ処理装置は、複数のオペレーティングシステムの要素の制御の下でのマルチプログラミング・モードで実行している場合があり、この場合、データ処理装置は、複数のノードとみなされる。典型的には、データ記憶装置は、データファイル、オペレーティング・システムとその情報管理要素、および、ユーザのアプリケーション・プログラムをストアしている。
【０００３】データは、企業にとっての重要な側面からビジネスを抽出した情報である。そのようなシステムでのチャレンジは、ビジネスニーズに合った使用可能性、パフォーマンスおよび費用でエンドユーザがデータにアクセスできる方法で、システムのデータ記憶と通信資源を使用することである。データへのアクセスは、データの整合性と完全性を確実にするために制御されなければならない。分散データ処理システム環境でのデータアクセスの付加的な特徴は、地理的および時間的親和性である。
【０００４】分散されたデータの構造の基礎は地理的親和性である。或るデータ項目へのアクセスは、地理的にまとまる傾向がある。データのダイナミックな複製方法の基礎は時間的親和性である。最近アクセスされたデータ項目は、最近アクセスされなかったデータ項目よりも、近い将来においてアクセスされる可能性が高い。或るデータ項目に対するアクセスがまとまる傾向があるノードは、親和性ノードと呼ばれる。或るデータ項目に対する親和性ノードは前もってわからず、また、そのノードは時間によって変化する。
【０００５】分散データ技術は、データのある場所（データロケーション）、データ共有（データシェアリング）の度合い、通信網全体にわたって提供されるデータベース制御の度合い、および、データアクセスの型の属性にしたがって分類することができる。データロケーションは、集中管理、分割化、または、複製化に分けられる。データシェアリングの度合いは、集中化、非集中化、または、分散化に分けられる。データベース管理の制御は、ユーザ提供のもの（分散データ）、または、システム提供のもの（分散データベース）に分けられる。データアクセスは、トランザクション処理型、機能処理型、または、データ処理型に分けられる。
【０００６】歴史的には、集中管理型がデータベース記憶とアクセスを管理するために使用されてきた。そのような処理手法においては、データ管理とアプリケーション処理が集中管理される。単一のデータベース・マネージャーが使用され、ユーザを中央システムに接続するためにテレプロセッシング通信網が使用される。集中処理手法の変形として、処理の或るものが、通信網の中のノードに分散される。しかし、データは集中管理される。
【０００７】データベースの集中管理手法の利点は、以下の通りである。（１）データベースの整合性が単一のデータベース・マネージャーによって確実にできること、（２）すべてのアプリケーション・プログラムが単一のアプリケーション・プログラミング・インターフェースにしたがって作成することができ、すべてのデータが１つのロケーションにストアされているので、アプリケーション・プログラムはデータロケーションを承知していなくてもよいこと、（３）集中データ処理環境でのデータ管理の問題を解決できる多くの有効なツールがあること、（４）単一システムは、操作し、維持し、制御するのがより容易であること。
【０００８】しかし、データベースの集中処理手法には、いくつかの不利な点がある。すなわち、（１）ある企業にとっては通信コストが高くつき、アプリケーション・プログラムの性能が通信遅延のためにさがること、（２）データ使用可能性が通信網あるいは中央システムの不安定性によって悪くなることがあり、これらの問題はバックアップ・システムや通信網に冗長度を持たせることによって解決されねばならないこと、（３）ある企業にとっては、単一の中央処理システムの処理能力がす

5

でに限界に達していること。

【０００９】分散データ処理システムのノードにデータを分散するには、一般的に２つの手法がある。これらの手法とは、分割化と静的複製化である。分割データ手法にはデータベースのプライマリ・コピーがないが、複製化手法にはデータベースのプライマリ・コピーがある。

【００１０】分割データベース手法は、データベースを明確な区画に分割し、これらが、ノードに分散される。したがって、あるデータ項目は、１つのノードにだけ存在する。各ロケーションは、そのロケーションでデータを管理するデータベース・マネージャーを持つ。データ分散マネジャーはアプリケーション・プログラムのデータ・リクエストを受け、データがそのロケーションすなわちローカルに存在する場合はそのリクエストをローカルリクエストにマップし、データが別のロケーションにある場合は、そのリクエストをリモートリクエストにマップする。

【００１１】分割された分散データベースでは、リクエストされたデータがローカルに存在する場合には、良いデータ使用可能性とアクセス性能が得られる。さらに、各データ項目が単一のデータベース・マネージャーによって管理されるので、データベースの整合性が容易に得られる。よい分割アルゴリズムが前もってわかり、それが存在し、且つ、安定したものであるならば、これらの結果を得ることができる。

【００１２】上述した分割されたデータベースでは、複数のロケーションでデータを変更するプログラムのために、システムが、通信網全体にわたる回復処理を提供しなければならない。

【００１３】分割されたデータベース・システムには、また、以下の不利な点がある。（１）分割アルゴリズムがデータアクセス・パターンに一致しない場合には、データの使用可能性と性能が落ちること、（２）アプリケーション・プログラムが、データロケーション、あるいは、少くともデータ分割アルゴリズムを承知していなければならず、また、データロケーションによってデータベースを異なった方法でアクセスしなければならないこと、（３）データベース分割アルゴリズムを変更することは、データロケーションが各ノードのアプリケーション・プログラムの中、エクシット、または、デクラレーションに反映されているので、非常に難しいこと、

（４）各ノードにおける既存データの再配置とアルゴリズムの変更は通信網全体にわたって同期して行われねばならず、したがって、最適な性能とデータの使用可能性を維持するのに必要な分割アルゴリズムを調整することができない場合があること、（５）分割されたデータベース全体にわたって或るデータ項目を均一にアクセスするか、または、全データベースをアクセスしなければならないプログラムは性能とデータ使用可能性の低下を蒙ること。

6

【００１４】データを分散するための静的複製手法は、中央ノードを含むか、または、含まない手法を含む。前者の場合、中央ロケーションはデータベースのプライマリ・コピーをストアし、各ロケーションはデータベース・マネジャーとデータベースのコピーを持つ。静的複製手法の典型的な使用方法では、プライマリ・データベースがコピーされて各レプリカ・ロケーションすなわちノードに送られ、各ノードでのローカル処理のためにデータが使用できるようになる。各レプリカ・ロケーションで行われたデータ変更は、プライマリ・データベースに対して後で処理を行うために集められる。アプリケーション・プログラムが処理を行う間をぬって、ローカルで行われたデータ変更が中央ロケーションに送られ、プライマリ・データベースに対してアップデートが行われる。レプリカ・データベースを管理するこの手法は複数回のアップデートを防ぐ手法を何も持たないので、プライマリ・データベースに対する各アップデートを手作業で発見して解決するか、あるいは、そのような複数回のアップデートが起こらないようにアプリケーション・プログラムを何らかの方法で制限しなければならない。プライマリ・データベースがレプリカ・コピーのアップデートに一致するようにした後、新しいコピーがレプリカ・ロケーションに送られ、全プロセスが再び開始される。

【００１５】プライマリ・コピーを中央ロケーションに持つ静的複製の主要な利点は、すべてのデータがローカルにアクセス可能であるので、高いデータ使用可能性と良い応答時間が得られる。しかし、この方法には以下の不利な点がある。（１）システムが複数回のアップデートを防ぐことができないのでデータベースの整合性を保つことが難しく、静的レプリカのために実行可能なデータベース処理が厳しく制限されること、（２）システムが、アプリケーション・プログラムのアクセスが必要とする最新データを保証できないこと、（３）オペレーション上の特別なプロシージャが、レプリカ・データの変更を集めプライマリ・データベースをアップデートするために必要になり、このプロシージャを実行することは高くつき、また、エラーを起こし易いこと、（４）データのアップデートの間にはデータ伝送のために不必要に大きいバンド幅を必要とし、確認のための十分大きなウィンドウを提供することは多くのアプリケーションでは実行不可能であり、レプリカがオペレーションの合間の狭いウィンドウの中だけで送られるので、確認プロシージャの間は、データベースが使用できなくなる可能性が大きいこと。さらに、１つあるいは複数のノードが動作不可能になった場合、予定されたウィンドウの中で確認プロシージャが行えなくなる。

【００１６】上述した静的複製の基本的な手法に関して、多くのさまざまな手法が文献に記載されている。たとえば、複数回のアップデートが起こらないようにアプ

リケーション・プログラムを設計することができるし、あるいは、レプリカは読み取りアクセスだけに制限することもできる。あるいは、アプリケーション・プログラム自身が、後でプライマリ・ロケーションに送るためにアップデートを集め、プライマリ・ロケーションにおいて、データベース・マネージャのログからこの情報を探り出すようにすることもできる。レプリカ・ロケーションでは、全面的なレプリカまたは部分的なレプリカだけを形成することもできる。レプリカ・データベース全体あるいは、保持されているデータへの変更のみを送るようにすることができる。トランザクションによって行われた変更を種々のノードに送り、トランザクション終了処理の1部としてアクノレッジメントを受領することによって、レプリカの同期性を常に保つようにすることができる。そのような同期手法によって静的複製の整合性の問題を解決するができる。しかし、これらの手法では、そのようなシステムの性能と使用可能性の利益の多くを失う。

【００１７】米国特許4,007,450号には、各ノードが他のノードと或る種のデータセットを共有し、中央ロケーションには絶えず同期されるレプリカだけがあるがプライマリ・コピーがない、分散データ制御システムが記載されている。各ノードは、他のノードのいずれかがアップデートしようとしていないかぎり、共有されたどのデータセットをもアップデートするように作動し、複数のノードがアップデートしようとしているときには、高いプライオリティーを持つノードがアップデートを行う。各ノードはそのメモリに、共有された各データセットのノードロケーションと、共有された各データセットに対して各ノードが持つアップデート・プライオリティーをストアしている。或るノードでデータセットがアップデートされるとき、レプリカを持っているすべてのノードにそのアップデートが送られる。上記のような手法は、静的複製の整合性問題を解決することはできるが、性能と使用可能性の利益の多くが失われる。

【００１８】米国特許4,432,057号には、分散マルチプロセシング・データベース・システムにおける資源の活用を制御するための分散システム制御の下での、データのダイナミックな複製方法が記載されている。このシステムでは、指定された現在性（currency）を持つデータへのアクセスリクエストが許され、アップデートされたデータの確認が、各ノードにおいて実行される制御プロシージャを使用することによって選択的に行われる。この場合、各ノードは、他ノードにおける共有データ項目のステータスに関するそのノードの見方をあらわし各ノードにファイルされているステータス・アンド・コントロール（status and control message:SAC）メッセージを使用する。このシステムでは、各データベース・リクエストはインターセプトされ、ローカル・コピーが現在性に対する要求を満たすかどうかが判断される。現在性が満たされない場合、そのリクエストを実行する前に、関連するノードと交渉を行うことが必要である。したがって、レプリカ・データベースの現在性は、アクセス・パターンに従ってダイナミックに変わる。

【００１９】上記のことから、データ処理システムの1つまたは複数のロケーションにおいて複製されるジャーナルされたデータベースを含む回復力のあるデータベース・システムで使用し、それにより、各レプリカ・データベース内で維持される整合性レベルを、任意に且つ選択的に割り当てることができる方法とシステムの必要性があることがわかる。

【００２０】

【発明が解決しようとする課題】本発明の目的は、分散データ処理システム内のデータベース・アクセス制御のための改善された方法を提供することである。

【００２１】さらに、本発明の目的は、分散データ処理システム内のプライマリ・データベースと1つまたは複数のレプリカ・データベースとの間の整合性を維持するための改善された方法とシステムを提供することである。

【００２２】さらに、本発明の目的は、分散データ処理システム内の回復可能なデータベース内部の複数のレプリカ・データベースに付す多様な整合性レベルをユーザが選ぶことができる、改善された方法とシステムを提供することである。

【００２３】

【課題を解決するための手段】上述の課題を達成する方法を以下に述べる。分散データ処理システム内の1つまたは複数のロケーションで実施されるジャーナルされたデータベースを含む回復可能なデータベース・システムにおいて、プライマリ・データベースとレプリカ・データベースとの間で維持される整合性レベルをそれぞれ記述する複数の多様な整合性レベルが指定される。ユーザは、各レプリカ・データベースのための具体的な整合性レベルを選択することができる。その後、プライマリ・データベースの中のレコードをアップデートすると、そのアップデートを使って、各レプリカ・データベース内の対応するレコードに対して、そのレプリカ・データベースに選択された整合性レベルと整合のとれる方法でアップデートが始まる。このようにして、プライマリ・データベースが故障した場合にもスムーズに切り替えが行えるように、プライマリ・データベースと完全に整合性のあるレプリカ・データベースを提供することもできるし、また、完全に整合性のあるデータを必要としないアプリケーションの照会に応答するためには第2のレプリカ・データベースを提供し、これにより、そのデータベースに対するアクセスの効率を大きく向上させることもできる。

【００２４】

【実施例】図1に、本発明の方法とシステムを実施する

のに使用できる、分散データ処理システム８の概略図を示す。分散データ処理システム８は、ローカルエリア・ネットワーク（以下ＬＡＮという）１０および３２のような複数のＬＡＮを持ち、各ＬＡＮはコンピュータ１２および３０のような複数の個々のコンピュータを持つことが望ましい。もちろん、当業者には明らかなように、各ネットワークには、ホストプロセサに接続された複数のインテリジェント・ワークステーションを使用することもできる。

【００２５】このようなデータ処理システムによく見られるように、各コンピューターは記憶装置１４および印刷出力装置１６を接続することができる。記憶装置１４は、本発明の方法とシステムにしたがい、分散データ処理システム８のユーザによって定期的にアクセスされ処理されるプライマリ・データベースまたはそのレプリカをストアするために使用される。公知の方法によって、そのようなプライマリ・データベースまたはそのレプリカは記憶装置１４にストアされ、データベースを維持しアップデートする責任を持つリソース・マネジャーまたはライブラリ・サービスと関連づけられる。

【００２６】図１はさらに、分散データ処理システム８が、メインフレーム・コンピュータ１８のような複数のメインフレーム・コンピュータを含み、それらが通信リンク２２によってＬＡＮ１０に接続されていることを示す。また、メインフレーム・コンピュータ１８は、ＬＡＮ１０の遠隔記憶装置としてはたらく記憶装置２０に接続されている。第２のＬＡＮ３２が、ゲートウェイ・サーバ２８への通信制御装置２６および通信リンク３４を介して、ＬＡＮ１０に接続されている。ゲートウェイ・サーバ２８は、ＬＡＮ３２をＬＡＮ１０につなげる役割をする別個のコンピュータあるいはインテリジェント・ワークステーションであることが望ましい。

【００２７】ＬＡＮ３２およびＬＡＮ１０に関連して述べたように、プライマリ・データベースまたはそのレプリカは記憶装置２０内にストアされ、そのようにストアされたプライマリ・データベースおよびそのレプリカに対するリソース・マネジャーまたはライブラリ・サービスの役割をするメインフレーム・コンピュータ１８によって制御される。

【００２８】もちろん、当業者には明らかなように、メインフレーム・コンピュータ１８はＬＡＮ１０から地理的に遠い距離に設置することができ、同様に、ＬＡＮ１０は、ＬＡＮ３２から相当な距離離れていてもよい。たとえば、ＬＡＮ３２がカリフォルニア州にあり、ＬＡＮ１０がテキサス州にあり、メインフレーム・コンピュータがニューヨークにあってもよい。

【００２９】上述したことからわかるように、分散データ処理ネットワーク８の或る１つの部分にいるユーザが、データ処理ネットワーク８の別の部分にストアされているデータベースにアクセスしたいことがしばしばある。そのようなクライアント／サーバ・システム環境では、データベース内にストアされたデータへのアクセスは、いわゆる「回復可能な（resilient）データベース・システム」を提供することによって増やすことができる。回復可能なデータベース・システムとは、分散データ処理システム内の１つまたは複数のロケーションにおいて複製化されている、ジャーナルされたデータベースである。そのようなデータベースでは、１つのデータベースがプライマリ・データベースとして、他の全てのデータベースはバックアップ・レプリカとして指定される。回復可能なデータベース・システムは、故障が起こったとき１つ前のポイントに回復する。すなわち、ここでいう故障とは、データベース・レプリカの１つが在るコンピュータの故障、あるいは、レプリカの１つを破壊するような媒体故障によって引き起こされる故障のことである。クラスター管理を行うことによって、アプリケーション・プログラマおよびエンドユーザに透明で、プライマリ・レプリカが故障したときにバックアップ・レプリカにスムーズに切り替えられる回復処理を提供することができる。

【００３０】高い使用可能性を持つデータベースを提供するためにデータベースが複製化されているシステムにおいては、そのようなシステムの自然な延長は、プライマリ・サーバのワークロードを軽減するために、ユーザがバックアップ・レプリカに照会できるようにすることである。しかし、或る整合性が強制されていない限り、レプリカ・データベースに対する照会トランザクションは、コミットしていないトランザクションによってセットされた値、あるいは、古い値を読んでしまう。あまり重要でないアプリケーションではこれでも良いかも知れないが、通常多くのアプリケーションは、プライマリ・データベースとの最低レベルの整合性を必要とする。

【００３１】このアプリケーションについて説明する目的で、ここでは、「アプリケーション」という用語は、ユーザによって始動されデータベース・オペレーションを呼び出すアプリケーション・コードのことをいう。データベース・オペレーションを呼び出すために使用される多くの多様なアプリケーションが現在使用可能であり、そのようなアプリケーションは本発明の方法とシステムにとって修正する必要はない。典型的には、そのようなアプリケーションがデータベース・オペレーションを呼び出すときはいつでも使える「アプリケーション・スタブ」も提供される。アプリケーション・スタブは呼び出しをトラップし、その呼び出しを特別ルーチンにまわす。このアプリケーション・スタブは一般的にクラスター管理によって提供され、その後、リクエストを、プライマリ・データベースを制御する機械にわたす。故障を回復するために必要などの回復ステップも、アプリケーション・スタブによってアプリケーションから隠されている。

【００３２】本発明の明細書で記述される「プライマリ・エージェント」はプライマリ・データベースを制御する機械の上にあり、データベース・オペレーションを実行するプロセスである。アプリケーション・プロセス１つにつき、１つのプライマリ・エージェントがある。本発明の方法とシステムで使用される「バックアップ・レシーバ」は、レプリカ・データベースへのアクセスを管理する機械の上にあり、プライマリ・データベースにほどこされた変更を記述するジャーナル・エントリを受け取り、それらのジャーナル・エントリをジャーナルのローカル・レプリカに記録するプロセスである。データベースのレプリカを制御する各機械の上の１つのデータベースに対して１つのバックアップ・レシーバ・プロセスがある。「バックアップ・アプライア」は、レプリカ・データベースを制御する機械の上にあり、データベースのローカル・レプリカにジャーナル・エントリを付すプロセスである。バックアップ・レプリカを制御する各機械の上の１つのデータベースに対して１つのバックアップ・アプライア・プロセスがある。

【００３３】図２に、本発明の方法とシステムにしたがい実施された回復可能なデータベース・システム内の、典型的な書き込みオペレーションの制御フローの概略を示す。図に示すように、ユーザ４０はアプリケーション４２を使用し、スタブ・ルーチン４４を呼び出し、プライマリ・データベース４６内でオペレーションを実行する。プライマリ・データベース４６を制御する機械の中のプライマリ・エージェント４８がこのオペレーションを実行する。この修正によって必要になったジャーナル・エントリがジャーナル５０内に挿入され、レプリカ・データベース５２内のバックアップ・レシーバ５４、および、レプリカ・データベース６０のような他のレプリカ・データベースに広げられる。

【００３４】バックアップ・レシーバ５４はそれらのジャーナル・エントリを受け取ったというアクノレッジメントを返し、その後、それらのジャーナル・エントリをそのジャーナルのローカルレプリカ（符号５８に示すような）内に、随時非同期的に入れる。バックアップ・アプライア５６は、ジャーナル５８内にあるそれらのジャーナル・エントリをレプリカ・データベース５２内の対応するレコードに付ける。これは、そのレコードをロックし、アップデートし、その後、アンロックすることによって行われる。この方法によって、レプリカ・データベース５２とプライマリ・データベース４６との間の整合性を保つことができる。

【００３５】したがって、バックアップ・アプライア５６により、ユーザが直接レプリカ・データベース５２からレコードを読んだ場合には部分的にしかアップデートされていないレコードを読むのを防ぐことによって、最低レベルの整合性を提供することができる。もちろん、データベース読取りオペレーションに対する制御フローは、図２に示したものに似ている。しかし、ジャーナル・エントリは生成されず、したがって、バクアップ・レシーバとバックアップ・アプライアは使用されない。図２に示した環境は、たとえば、ＩＢＭ　ＡＳ／４００のようなコンピュータ・システムを使用して実施することができる。

【００３６】次に、本発明の重要な特徴として、選択されたレプリカ・データベースをユーザが使用する際にユーザが選択する複数の多様な整合性レベルが提供される。回復可能なデータベース・システム内の同時実行については、「同時実行制御とデータベース・システムの回復」（"Concurrence Control and Recovery in Database Systems" Addison-Wesley Publishing Company, 1987）に記載があり、複数バージョンの同時実行制御モデルが記述されている。このシステムにおいては、バックアップ・レプリカ・データベースは常にプライマリ・データベースより遅れ、したがって、データベースのバージョンが２つできる。上記文献のシステムに対比して、本発明の方法とシステムは、考慮するべき２つの異なる整合性属性を提供する。

【００３７】具体的には、これらの２つの属性は、データベース・システム内のトランザクションの順序付け（ordering）と、これらのトランザクションの直列化可能性（serializability）を含む。これらの２つの属性はいくつかの方法で組み合わせることができ、その結果６つの異なる整合性レベルができる。したがって、本発明の重要な特徴にしたがい、レプリカ・データベースから読み込む照会に対し、トランザクションの順序付けに関して３つの異なる整合性レベルが考慮される。もちろん、当業者には明らかなように、レコード・アップデートの後の整合性維持のために、最大時間遅れの値を指定することができる。「レベル０」整合性レベルは、レプリカ・データベース内の古い値を使って照会することを許す。「レベル１」整合性レベルは、レプリカ・データベースから最新のコミットされた値だけを使って照会できるという整合性を課す。最後に、「レベル２」整合性レベルは、その値がコミットに達したか否かに拘わらず、レプリカ・データベースからの最新の値だけを使って照会するという整合性を課す。

【００３８】次に、上述したように、所望の整合性レベルを割り当てるために、本発明の回復可能なデータベース・システム内のトランザクションの直列化可能性について考慮しなければならない。この属性によって、現実には同時にインターリーブして実行されたトランザクションが、或る直列順序で実行されたのと論理的には等しい効果を持たせることを保証する。回復可能なデータベース・システムは、コミットメント制御の下で動くアプリケーションが、コミットメント制御の下にない他のアプリケーションと同時に同じデータベースに対して動くことを許す。したがって、異なるアプリケーションは、

直列化可能性を必要としてもしなくても良いことが考えられる。したがって、本発明の回復可能なデータベース・システム内のレプリカ・データベースにユーザまたはシステム管理者が異なる多様な整合性レベルをセットすることを許すとき、整合性レベルを制御するために２つの異なる変数がセットされなければならない。順序付けは、「レベル０」、「レベル１」、あるいは「レベル２」にセットされる。さらに、直列化可能性は「必要」あるいは「不要」にセットされる。

【００３９】図３に、本発明の方法とシステムによる回復可能なデータベース内での複数の多様な整合性レベルのセットを説明する論理フローの概略図を示す。図示したように、このプロセスは、ブロック７０で始まり、ブロック７２に進む。ブロック７２で、整合性レベルの選択が行われたかどうかが判断される。Ｎｏの場合、このプロセスは、ユーザまたはシステム管理者が回復可能なデータベース・システム内のレプリカ・データベースの整合性レベルをセットしたいという意志表示を示すまで、繰り返される。

【００４０】ブロック７２で整合性レベルの選択が行われた場合、プロセスはブロック７４に進む。ブロック７４は、本発明の回復可能なデータベース・システム内の最初のレプリカ・データベースに対するプロセスを示す。次にプロセスはブロック７６に進む。ブロック７６で、プロセスは、ユーザあるいはシステム・オペレータが所望の整合性レベルをセットすることを促す。次に、プロセスはブロック７８に進む。ブロック７８で、所望の整合性レベルが入れられたかどうかが判断され、Ｎｏの場合は、プロセスは繰り返しブロック７６に戻り、再度所望の整合性レベルを入れるように促す。

【００４１】ブロック７８で所望の整合性レベルが入れられたと判断されれば、プロセスはブロック８０に進む。ブロック８０で、直列化可能性の要求があるかどうかをユーザあるいはシステム・オペレータに促す。すなわち、インターリーブされて実行されたトランザクションが、或る直列順序で実行されたのと論理的に等しいものとしたいかどうかを、ユーザあるいはシステム・オペレータに意志表示してもらう。次に、プロセスはブロック８２に進む。ブロック８２で、ユーザが直列化可能性要求を入れたかどうかを判断し、Ｎｏの場合は繰り返しブロック８０に戻り、再度、このデータベースに対する直列化可能性要求があるか否かを入れるようにユーザに促す。ユーザが直列化可能性要求を入れた後、プロセスはブロック８２からブロック８４に進む。ブロック８４で、整合性レベルを付けるべきレプリカ・データベースがまだあるかどうかを判断し、以上のプロセスで整合性レベルを付けたレプリカ・データベースが最後のレプリカでない場合には、プロセスは繰り返しブロック７６に戻り、再度上述したプロセスを実行する。ブロック８４がＹｅｓの場合、すなわち、最後のレプリカ・データベースに所望の整合性レベルが付けられた場合は、プロセスはブロック８６に進みリターンする。

【００４２】図４から図１０に、本発明の方法とシステムによる回復可能なデータベース・システム内で、レコード・アップデートが行われた後の、複数の多様な整合性レベルを実行する方法を示すフローチャートの概略図を示す。図４に示すように、プロセスはブロック９０から始まる。次に、プロセスはブロック９２に進み、レコード・アップデートが起こったかどうかを判断する。レコード・アップデートがまだ起こっていない場合、このプロセスは、プライマリ・データベースにレコード・アップデートが起こるまで繰り返される。次に、レコード・アップデートが起こったときに、プロセスはブロック９２からブロック９４に進む。

【００４３】ブロック９４は、本発明の回復可能なデータベース・システム内の各レプリカ・データベースに対して繰り返して行われるプロセスの始まりを示す。プロセスはブロック９４からブロック９６に進む。ブロック９６で、所望の整合性レベルが「レベル０」にセットされたかどうかが判断される。Ｙｅｓであれば、プロセスはブロック９８に進む。ブロック９８で直列化可能性が要求されているかどうかが判断され、Ｙｅｓの場合、プロセスは、図４のコネクタ１００を介して図５の論理フローに進む。そうでない場合、すなわち、直列化可能性が不要であった場合、プロセスは、図４のコネクタ１０２を介して図６の論理フローに進む。

【００４４】再びブロック９６に戻り、所望の整合性レベルが「レベル０」にセットされなかった場合は、プロセスはブロック１０４に進む。ブロック１０４で、所望の整合性レベルが「レベル１」にセットされたかどうかが判断される。Ｙｅｓであれば、プロセスはブロック１０４からブロック１０６に進み、再度、直列化可能性が要求されているかどうかが判断される。直列化可能性が要求されている場合には、プロセスは、図４のコネクタ１０８を介して図７の論理フローに進む。そうでない場合、すなわち、直列化可能性が不要であった場合、コネクタ１１０を介して図８の論理フローに進む。

【００４５】再びブロック１０４に戻り、所望の整合性レベルが「レベル１」でない場合、プロセスはブロック１０４からブロック１１２に進む。ブロック１１２で、このレプリカ・データベースに対する所望の整合性レベルが「レベル２」にセットされたかどうかが判断され、もしＹｅｓであれば、プロセスはブロック１１２からブロック１１４に進む。前と同じように、ブロック１１４で直列化可能性が要求されたかどうかが判断され、Ｙｅｓの場合、プロセスはコネクタ１１６を介して図９の論理フローに進む。Ｎｏの場合、すなわち、直列化可能性が不要であった場合、プロセスはブロック１１４からコネクタ１１８を介して図１０の論理フローに進む。最後に、再びブロック１１２で、所望の整合性レベルが「レ

ベル２」でない場合、プロセスはブロック１２０に進みリターンする。

【００４６】図５に、レプリカ・データベース内で整合性「レベル０」が選択され直列化可能性が要求されたレコードに対応するレコードにレコード・アップデートが行われた場合に起こる論理フローの順序を示す。図５から図１０のそれぞれには、番号１３０、１３２、１３４、および、１３６を付けた４つのカラムがある。それぞれの場合において、カラム１３０はプライマリ・データベースに起こるアクティビティを示す。カラム１３２はジャーナル・エントリのフローを示し、カラム１３４は考慮対象の特定のレプリカ・データベース内のバックアップ・レシーバに起こるアクティビティを示す。最後に、カラム１３６は、レプリカ・データベース内のバックアップ・アプライア（図２を参照）のアクティビティを示す。

【００４７】したがって、図５のカラム１３０を見ると、ブロック１５０に、レコードＲ１に対するアップデートＵ１を含むトランザクションＴ１が示されている。この結果ジャーナル・エントリ（ＪＥ１）が作られ、これがバックアップ・レシーバに送られる。次に、バックアップ・レシーバはアクノレッジメントをプライマリ・データベースに送り、ジャーナル・エントリ（ＪＥ１）を、レプリカ・データベースにあるローカルジャーナルに入れる。次に、バックアップ・アプライアはレコードＲ１をロックし、ジャーナル・エントリ（ＪＥ１）を付ける。

【００４８】次に、ブロック１５２に、レコードＲ２に対するアップデートＵ２を含むトランザクションＴ２が示されている。この結果ジャーナル・エントリ（ＪＥ２）が作られ、これがバックアップ・レシーバに送られる。次に、バックアップ・レシーバはアクノレッジメントをプライマリ・データベースに送り、ジャーナル・エントリ（ＪＥ２）をローカル・ジャーナルに入れる。その後、バックアップ・アプライアはレコードＲ２をロックし、ジャーナル・エントリ（ＪＥ２）を付ける。

【００４９】次に、ブロック１５４に、レコードＲ３に対するアップデートＵ３を含むトランザクションＴ１を示す。このトランザクションの結果第３のジャーナル・エントリ（ＪＥ３）が作られ、これがバックアップ・レシーバに送られる。バックアップ・レシーバは再度アクノレッジメントを送り、このジャーナル・エントリをローカルジャーナルに入れる。次に、バックアップ・アプライアはレコードＲ３をロックし、ジャーナル・エントリ（ＪＥ３）を付ける。

【００５０】最後に、ブロック１５６に、トランザクションＴ１のためのコミット・オペレーションを示す。このトランザクションの結果、ジャーナル・エントリ（ＪＥ４）が作られ、バックアップ・レシーバに送られる。バックアップ・レシーバはアクノレッジメントをプライマリ・データベースに送り、ジャーナル・エントリ（ＪＥ４）をローカルジャーナルに入れる。次に、バックアップ・アプライアはレコードＲ１とＲ３に対するロックをリリースする。

【００５１】図６は、整合性レベルを「レベル０」にセットし直列化可能性を不要にしたレプリカ・データベース内のレコードにアップデートをした結果起こる論理フローを示す。説明を簡単にするために、図６から図１０のカラムおよびトランザクションは、図５に使ったものと同じ番号を使用している。上述したのと同じように、プロセスは、レコードＲ１に対するアップデートＵ１を含むトランザクションＴ１を示すブロック１５０から始まる。このトランザクションの結果、ジャーナル・エントリ（ＪＥ１）が作られ、これがバックアップ・レシーバに結合される。アクノレッジメントがバックアップ・レシーバからプライマリ・データベースに送られ、ジャーナル・エントリ（ＪＥ１）がローカル・ジャーナルに入れられる。次に、バックアップ・アプライアはレコードＲ１をロックし、ジャーナル・エントリ（ＪＥ１）を付ける。その後、直列化可能性が不要なので、バックアップ・アプライアはレコードＲ１をリリースする。

【００５２】ブロック１５２に、レコードＲ２に対するアップデートＵ２を含むトランザクションＴ２を示す。このトランザクションの結果ジャーナル・エントリ（ＪＥ２）が作られ、これがバックアップ・レシーバに送られる。バックアップ・レシーバはアクノレッジメントをプライマリ・データベースに送り、ジャーナル・エントリ（ＪＥ２）をローカルジャーナルに入れる。次に、バックアップ・アプライアはレコードＲ２をロックし、ジャーナル・エントリ（ＪＥ２）を付け、その後、レコードＲ２をリリースする。

【００５３】ブロック１５４に、レコードＲ３に対するアップデートＵ３を含むトランザクションＴ１の追加部分を示す。このトランザクションの結果ジャーナル・エントリ（ＪＥ３）が作られ、バックアップ・レシーバに送られる。バックアップ・レシーバはアクノレッジメントをプライマリ・データベースに送り、ジャーナル・エントリ（ＪＥ３）をローカルジャーナルに入れる。次に、バックアップ・アプライアはレコードＲ３をロックし、ジャーテル・エントリ（ＪＥ３）を付け、その後、レコードＲ３をリリースする。

【００５４】最後に、ブロック１５６に、トランザクションＴ１に対するコミット・オペレーションを示す。このコミット・オペレーションの結果、ジャーナル・エントリ（ＪＥ４）が作られ、バックアップ・レシーバに送られる。アクノレッジメントがバックアップ・レシーバによりプライマリ・データベース送られ、ジャーナル・エントリ（ＪＥ４）がローカルジャーナルに入れられる。直列化可能性が不要であるので、このコミット・オペレーションの結果、バックアップ・アプライアによる

アクティビティは起こらない。

【００５５】次に、図７に、レプリカ・データベースに対して整合性「レベル１」が選択され直列化可能性が要求されている場合のプライマリ・データベース内のレコードに対するアップデートへの応答の方法を示す論理フローを示す。前と同じように、ブロック１５０は、レコードＲ１に対するアップデートＵ１を含むトランザクションＴ１を示す。このトランザクションの結果ジャーナル・エントリ（ＪＥ１）が作られ、バックアップ・レシーバに送られる。アクノレッジメントがバックアップ・レシーバによってプライマリ・データベースに送られ、ジャーナル・エントリ（ＪＥ１）がローカルジャーナルに入れられる。次に、バックアップ・アプライアはこのジャーナル・エントリ（ＪＥ１）をバッファにストアする。上述したように、整合性「レベル１」は、照会がコミットされた最新の値のみを読むことを課しているので、当業者には明らかなように、レコードＲ１に対するアップデートは、コミット・トランザクションが起こるまでの間、必ずバッファにストアされなければならない。

【００５６】次に、ブロック１５２に示すように、レコードＲ２に対するアップデートＵ２を含むトランザクションＴ２が起こる。この結果ジャーナル・エントリ（ＪＥ２）が作られ、バックアップ・レシーバに結合される。バックアップ・レシーバは、アクノレッジメントをプライマリ・データベースに送り、ジャーナル・エントリ（ＪＥ２）をローカルジャーナルに入れる。前と同じように、次に、バックアップ・アプライアはこのジャーナル・エントリ（ＪＥ２）をバッファにストアする。

【００５７】次に、ブロック１５４に示すように、レコードＲ３に対するアップデートＵ３を含むトランザクションＴ１が起こる。この結果ジャーナル・エントリ（ＪＥ３）が作られ、バックアップ・レシーバに結合される。前と同じように、アクノレッジメントがバックアップ・レシーバからプライマリ・データベースに送られ、ジャーナル・エントリ（ＪＥ３）がローカルジャーナルに入れられる。バックアップ・アプライアはこのジャーナル・エントリ（ＪＥ３）をバッファ内にストアする。

【００５８】最後に、ブロック１５６に示すように、トランザクションＴ１のためのコミット・オペレーションが起こる。この結果、ジャーナル・エントリ（ＪＥ４）が作られ、バックアップ・レシーバに送られる。バックアップ・レシーバはレコードＲ１とＲ３をロックし、アクノレッジメントをプライマリ・データベースに送る。その後、バックアップ・アプライアはトランザクションＴ１からジャーナル・エントリを取り出し、該当するジャーナル・エントリ、すなわち、トランザクションＴ１に関連しているジャーナル・エントリ（ＪＥ１およびＪＥ３）を付ける。その後、レコードＲ１およびＲ３はリリースされる。したがって、レコードＲ１およびＲ３に対するアップデートは、そのトランザクションのためのコミット・オペレーションが起こるまで、バックアップ・アプライアによって適用されない。

【００５９】図８に、整合性レベルを「レベル１」に、直列化可能性を不要にセットしたレプリカ・データベース内でアップデートされたレコードに対する応答方法を記述した論理フローを示す。前と同じように、ブロック１５０は、レコードＲ１に対するアップデートＵ１を含むトランザクションＴ１を示す。この結果、ジャーナル・エントリ（ＪＥ１）が作られ、バックアップ・レシーバに結合される。バックアップ・レシーバはアクノレッジメントをプライマリ・データベースに送り、このジャーナル・エントリ（ＪＥ１）をローカルジャーナルに入れる。次に、ジャーナル・エントリ（ＪＥ１）は、バックアップ・アプライアによってバッファにストアされる。

【００６０】次に、ブロック１５２に示すように、レコードＲ２に対するアップデートＵ２を含むトランザクションＴ２が起こる。この結果、ジャーナル・エントリ（ＪＥ２）が作られ、バックアップ・レシーバに結合される。次に、バックアップ・レシーバはアクノレッジメントをプライマリ・データベースに送り、このジャーナル・エントリ（ＪＥ２）をローカルジャーナルに入れる。次に、前と同じように、このジャーナル・エントリはバックアップ・アプライアによってバッファにストアされる。

【００６１】ブロック１５４で、レコードＲ３に対するアップデートＵ３を含むトランザクションＴ１が起こる。この結果ジャーナル・エントリ（ＪＥ３）が作られ、バックアップ・レシーバに結合される。アクノレッジメントがバックアップ・レシーバからプライマリ・データベースに送られ、ジャーナル・エントリ（ＪＥ３）がローカルジャーナル内にストアされる。次に、前と同じように、ジャーナル・エントリ（ＪＥ３）がバックアップ・アプライアによってバッファにストアされる。

【００６２】最後に、ブロック１５６に示すように、トランザクションＴ１のためのコミット・オペレーションが起こる。この結果ジャーナル・エントリ（ＪＥ４）が作られ、バックアップ・レシーバに結合される。次に、バックアップ・レシーバはレコードＲ１とＲ３をロックし、アクノレッジメントをプライマリ・データベースに送る。その後、バックアップ・アプライアはトランザクションＴ１に対するジャーナル・エントリを取り出し、第１のジャーナル・エントリ（ＪＥ１）を適用する。次に、レコードＲ１がリリースされる。次に、トランザクションＴ１に対する第２のジャーナル・エントリ（ＪＥ３）がレコードＲ３に適用され、そのレコードはリリースされる。図７の論理フローとは対照的に、図８の論理フローでは直列化可能性の要求がないので、各レコードはそのアップデートが適用された後でリリースされる点

に留意する必要がある。

【００６３】次に、図９に、整合性レベルを「レベル２」にセットし直列化可能性を必要とするプライマリ・データベース内の対応するレコードのアップデートに対する、レプリカ・データベースの応答方法を記述した論理フローを示す。前と同じように、ブロック１５０で、レコードＲ１に対するアップデートＵ１を含むトランザクションＴ１が起こる。このトランザクションの結果、ジャーナル・エントリ（ＪＥ１）が作られ、バックアップ・レシーバに送られる。次に、バックアップ・レシーバはＲ１をロックし、アクノレッジメントをプライマリ・データベースに送る。次に、ジャーナル・エントリ（ＪＥ１）がローカルジャーナルに入れられ、バックアップ・アプライアは直ちにジャーナル・エントリ（ＪＥ１）を適用する。当業者には明らかなように、整合性「レベル２」は、照会は最新の値（コミットが起こったかどうかに拘わらず）のみを読むことによって行うことを課している。したがって、レコードＲ１に対するアップデートは、バックアップ・アプライアによって直ちに適用される。

【００６４】ブロック１５２で、レコードＲ２に対するアップデートＵ２を含むトランザクションＴ２が起こる。この結果、ジャーナル・エントリ（ＪＥ２）が作られ、バックアップ・レシーバに結合される。次に、バックアップ・レシーバはレコードＲ２をロックし、アクノレッジメントをプライマリ・データベースに送り、ジャーナル・エントリ（ＪＥ２）をローカルジャーナルに入れる。次に、バックアップ・アプライアは直ちにジャーナル・エントリ（ＪＥ２）を適用し、レコードＲ２を最新の値にアップデートする。

【００６５】ブロック１５４で、レコードＲ３に対するアップデートＵ３を含むトランザクションＴ１の第２部分が起こる。この結果、ジャーナル・エントリ（ＪＥ３）が作られてバックアップ・レシーバに結合される。次に、バックアップ・レシーバはレコードＲ３をロックし、アクノレッジメントをプライマリ・データベースに送る。次に、ジャーナル・エントリ（ＪＥ３）がローカルジャーナルに入れられ、バックアップ・アプライアはこのジャーナル・エントリ（ＪＥ３）をレコードＲ３に適用する。

【００６６】最後に、ブロック１５６で、トランザクションＴ１に対するコミット・オペレーションが起こる。この結果、ジャーナル・エントリ（ＪＥ４）が作られ、バックアップ・レシーバに結合される。次に、バックアップ・レシーバはアクノレッジメントをプライマリ・データベースに送り、このジャーナル・エントリ（ＪＥ４）をローカルジャーナルに入れる。次に、バックアップ・アプライアはレコードＲ１とＲ３をリリースし、必要に応じて、このトランザクション部分の直列化可能性を維持する。

【００６７】最後に、図１０に、整合性「レベル２」をセットし直列化可能性を要求しないレプリカ・データベースの中のプライマリ・データベース内のレコードに対するアップデートに応答して起こる論理フローを示す。前と同様に、ブロック１５０で、レコードＲ１に対するアップデートＵ１を含むトランザクションＴ１が起こる。この結果、ジャーナル・エントリ（ＪＥ１）が作られバックアップ・レシーバに結合される。次に、バックアップ・レシーバはレコードＲ１をロックし、アクノレッジメントをプライマリ・データベースに送る。次に、ジャーナル・エントリ（ＪＥ１）がローカルジャーナルに入れられる。次に、バックアップ・アプライアはジャーナル・エントリ（ＪＥ１）を適用し、直列化可能性が要求されていないので、直ちにレコードＲ１をリリースする。

【００６８】次に、ブロック１５２に示すように、レコードＲ２に対するアップデートＵ２を含むトランザクションＴ２が起こる。この結果、ジャーナル・エントリ（ＪＥ２）が作られバックアップ・レシーバに結合される。次に、バックアップ・レシーバはレコードＲ２をロックし、アクノレッジメントをプライマリ・データベースに送る。次に、ジャーナル・エントリ（ＪＥ２）がローカルジャーナルに入れられる。次に、バックアップ・アプライアはジャーナル・エントリ（ＪＥ２）を適用し、レコードＲ２をリリースする。

【００６９】次に、ブロック１５４に示すように、レコードＲ３に対するアップデートＵ３を含むトランザクションＴ１の次の部分が起こる。この結果、ジャーナル・エントリ（ＪＥ３）が作られ、バックアップ・レシーバに結合される。次に、バックアップ・レシーバはレコードＲ３をロックし、アクノレッジメントをプライマリ・データベースに送る。次に、ジャーナル・エントリ（ＪＥ３）がローカルジャーナルに入れられ、バックアップ・アプライアはジャーナル・エントリ（ＪＥ３）を適用し、レコードＲ３をリリースする。以上の記述から当業者には明らかなように、この整合性レベルでは直列化可能性が要求されていないので、レコードＲ１およびＲ３に対するアップデートが行われ、各レコードは、そのアップデートが行われた後直ちにリリースされる。

【００７０】最後に、ブロック１５６で、トランザクションＴ１に対するコミット・オペレーションが起こる。この結果、ジャーナル・エントリ（ＪＥ４）が作られ、バックアップ・レシーバに結合される。次に、バックアップ・レシーバはこのジャーナル・エントリのアクノレッジメントを送り、ジャーナル・エントリ（ＪＥ４）をローカルジャーナルに入れる。

【００７１】まとめとして、本発明の構成に関して以下の事項を開示する。

（１）分散データ処理システム内でのデータベース・アクセス効率を向上させる方法であって、前記分散データ

処理システム内において、ストアされた複数のレコードを持つ１つのデータベースを選択しプライマリ・データベースとして指定するステップと、前記分散データ処理システム内の第２の物理的ロケーションにおいて前記プライマリ・データベースを複製するステップと、前記プライマリ・データベースと前記の複製されたレプリカ・データベースとの間に維持される複数の多様な整合性レベルを指定するステップと、ユーザに前記の複数の多様な整合性レベルから１つを指定させるステップと、を有し、前記プライマリ・データベース内のレコードに対するアップデートに応答して、前記プライマリ・データベースと前記レプリカ・データベースとの間に前記の複数の多様な整合性レベルの前記の指定された１つを自動的に維持する前記方法。

（２）前記分散データ処理システム内の第３の物理的ロケーションにおいて前記プライマリ・データベースを複製するステップをさらに有する、前記（１）に記載の方法。

（３）ユーザに前記の複数の多様な整合性レベルから１つを指定させる前記ステップが、前記プライマリ・データベースとそのそれぞれのレプリカとの間に維持される前記の複数の多様な整合性レベルから別の１つを指定させるステップを有する、前記（２）に記載の方法。

（４）分散データ処理システム内でのデータベース・アクセス効率を向上させるシステムであって、前記分散データ処理システム内において、ストアされた複数のレコードを持つプライマリ・データベースと、前記分散データ処理システム内の第２の物理的ロケーションにおいてストアされている前記プライマリ・データベースのレプリカと、前記プライマリ・データベースと前記レプリカ・データベースとの間に維持される複数の多様な整合性レベルを指定する手段と、ユーザに前記の複数の多様な整合性レベルから１つを指定させる手段と、前記プライマリ・データベース内のレコードに対するアップデートに応答して、前記プライマリ・データベースと前記レプリカ・データベースとの間に前記の複数の多様な整合性レベルの前記の指定された１つを自動的に維持する手段と、を有するシステム。

（５）前記分散データ処理システム内の第３の物理的ロケーションにおいてストアされている前記プライマリ・データベースの第２のレプリカをさらに有する、前記（４）に記載のシステム。

（６）ユーザに前記の複数の多様な整合性レベルから１つを指定させる前記手段が、前記プライマリ・データベースとそのそれぞれのレプリカとの間に維持される前記の複数の多様な整合性レベルから別の１つを指定させる手段を有する、前記（５）に記載のシステム。

（７）分散データ処理システム内でのデータベース・アクセス効率を向上させる方法であって、前記分散データ処理システム内において、ストアされた複数のレコード

を持つ１つのデータベースを選択しプライマリ・データベースとして指定するステップと、前記分散データ処理システム内の第２の物理的ロケーションにおいて前記プライマリ・データベースを複製するステップと、前記プライマリ・データベース内の１つの選択されたレコードに対するアップデートに応答して、前記の複製されたデータベース内の対応する選択されたレコードへのアクセスを自動的にロックするステップと、前記の複製されたデータベース内の前記の選択されたレコードをアップデートするステップと、前記アップデートの後で、前記の選択されたレコードへのアクセスをリリースするステップと、前記分散データ処理システム内のユーザに、前記プライマリ・データベースおよび前記の複製されたデータベースの中のレコードを照会させるステップと、を有し、データの絶対的な整合性を維持するとともにデータアクセスの使用可能性を向上させる方法。

（８）前記プライマリ・データベース内のレコードに対する各アップデートが、コミット・オペレーションを行う前の１つの整合性時点に戻り、前記の選択されたレコードへのアクセスを前記アップデートの後でリリースする前記ステップが、前記アップデートに続くコミット・オペレーションの後でのみ前記の選択されたレコードへのアクセスをリリースするステップを有する、前記（７）に記載の方法。

（９）分散データ処理システム内でのデータベース・アクセス効率を向上させるシステムであって、前記分散データ処理システム内において、ストアされた複数のレコードを持つプライマリ・データベースと、前記分散データ処理システム内の第２の物理的ロケーションにおいてストアされている前記プライマリ・データベースのレプリカと、前記プライマリ・データベース内の１つの選択されたレコードに対するアップデートに応答して、前記の複製されたデータベース内の対応する選択されたレコードへのアクセスを自動的にロックする手段と、前記の複製されたデータベース内の前記の選択されたレコードをアップデートする手段と、前記アップデートの後で、前記の選択されたレコードへのアクセスをリリースする手段と、前記分散データ処理システム内のユーザに、前記プライマリ・データベースおよび前記の複製されたデータベースの中のレコードを照会させる手段と、を有し、データの絶対的な整合性を維持するとともにデータアクセスの使用可能性を向上させるシステム。

（１０）前記プライマリ・データベースを、いかなる時点でも、コミット・オペレーションを行う前の整合性時点に戻す手段をさらに有する、前記（９）に記載のシステム。

（１１）前記アップデートの後で前記の選択されたレコードへのアクセスをリリースする前記手段が、前記アップデートに続くコミット・オペレーションの後でのみ前記の選択されたレコードへのアクセスをリリースする手

段を有する、前記（１０）に記載のシステム。

【００７２】

【発明の効果】本発明の方法とシステムによって、ユーザあるいはシステム・オペレータは、バックアップ・サーバ内のレプリカ・データベースに適する任意の整合性レベルを広範な整合性レベルから選択できるようになる（この選択には、整合性と性能との間のトレードオフが含まれる）。この方法とシステムにより、最低限の整合性レベルでは、１つのレプリカ・データベースにおいて、古い、コミットされていない値を読むことができる一方、他方では、プライマリ・データベースの第２のレプリカ内で、あたかも、照会がプライマリ・データベースから読み込みを行ったのと同じ結果を返すことができる最大レベルの整合性を提供することもできる。このようにして、上述した回復可能なデータベース・システム内において、データベースへのアクセスの効率を大いに改善することができる。

【図面の簡単な説明】

【図１】本発明の方法およびシステムを実施するのに使用できる分散データ処理システムの概略図である。

【図２】本発明の方法およびシステムにしたがって実施される回復可能なデータベース・システム内での、典型的な書き込みオペレーションの概略制御フローを示す。

【図３】本発明の方法およびシステムによる回復可能なデータベース内での複数の多様な整合性レベルの設定を図示する概略論理フローチャートである。

【図４】本発明による複数の多様な整合性レベルの実施を図示する概略論理フローチャートであり、図５に続く。

【図５】図４に続く論理フローチャート。

【図６】図５に続く論理フローチャート。

【図７】図６に続く論理フローチャート。

【図８】図７に続く論理フローチャート。

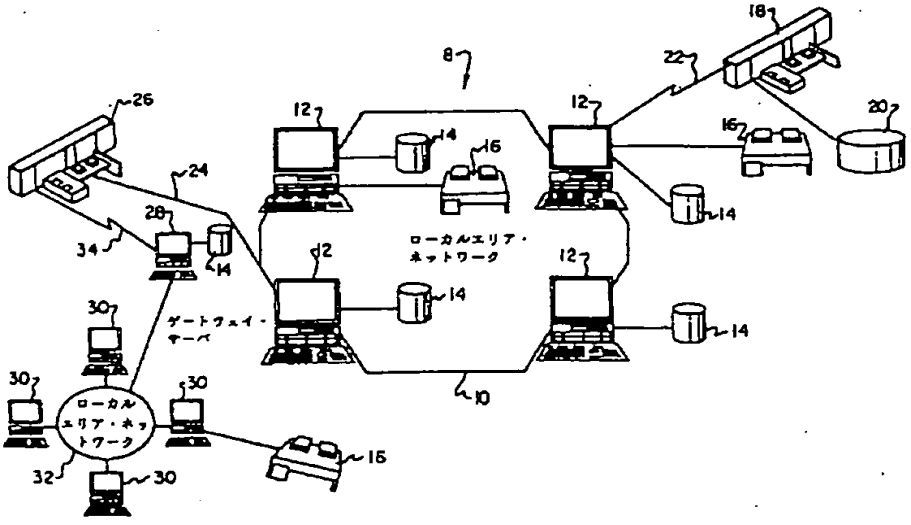
【図９】図８に続く論理フローチャート。

【図１０】図９に続く論理フローチャート。

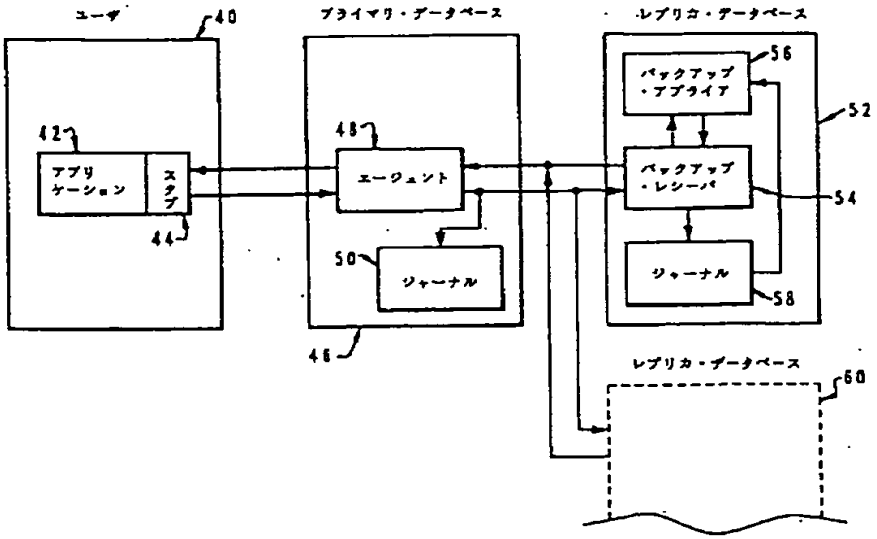
【符号の説明】

８	分散データ処理システム
１０、３２	ローカルエリア・ネットワーク（ＬＡＮ）
１２、３０	コンピュータ
１４、２０	記憶装置
１６	印刷出力装置
１８	メインフレーム・コンピュータ
２２、２４、３４	通信リンク
２６	通信制御装置
２８	ゲートウェイ・サーバ
４０	ユーザ
４２	アプリケーション
４４	スタブ・ルーチン
４６	プライマリ・データベース
４８	プライマリ・エージェント
５０、５８	ジャーナル
５２	レプリカ・データベース
５４、６０	バックアップ・レシーバ
５６	バックアップ・アプライア

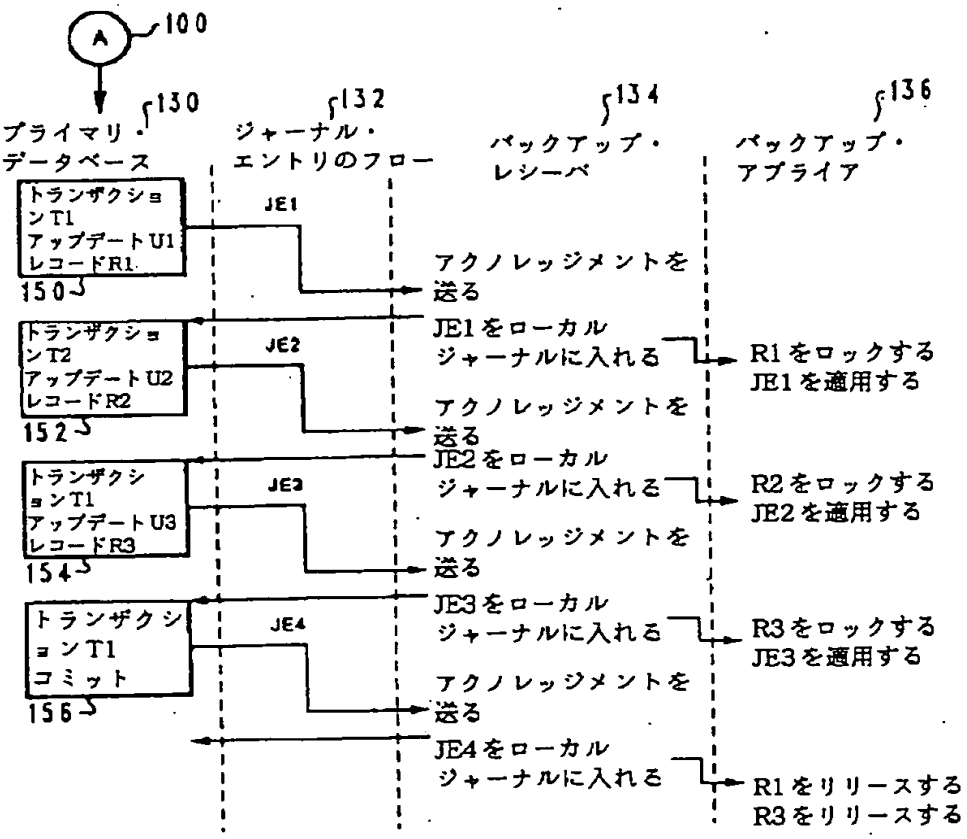
【図１】



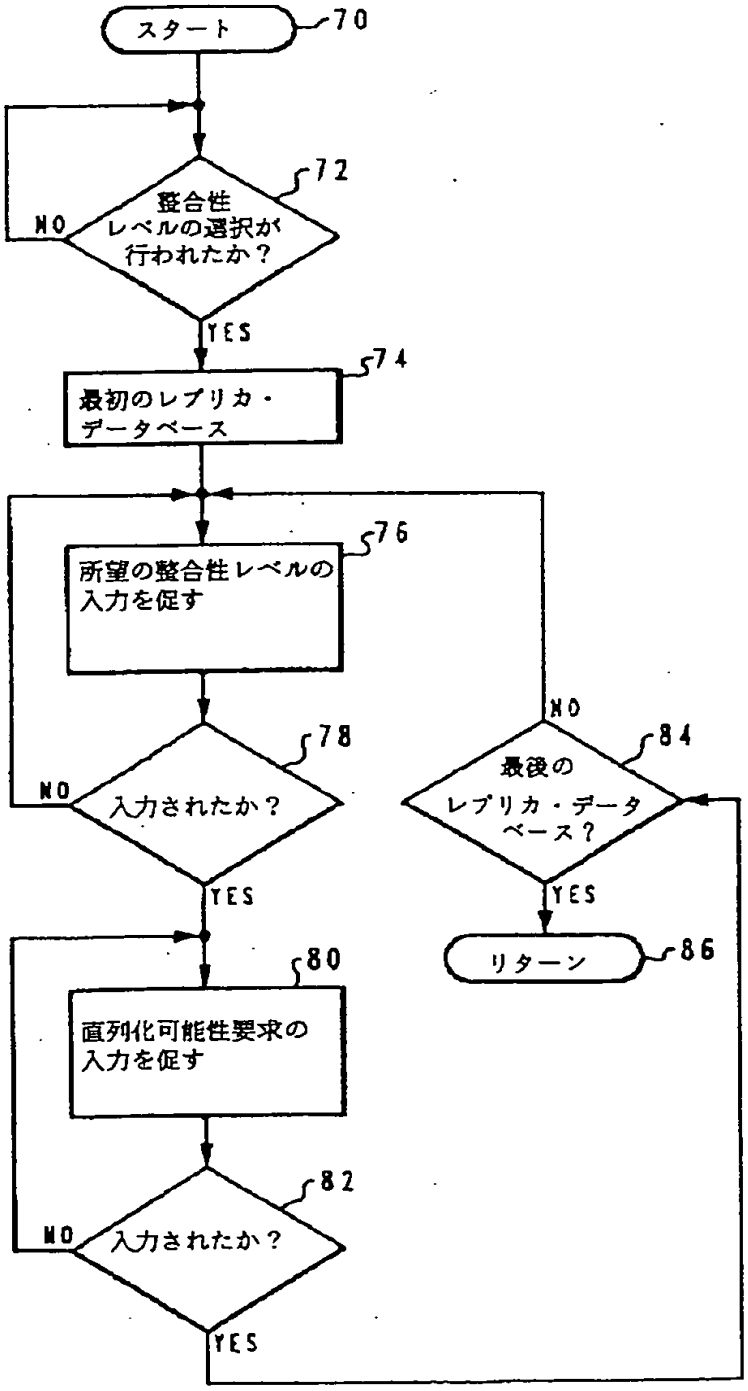
【図２】


ユーザ
40
プライマリ・データベース
レプリカ・データベース
42
アプリケーション
スタブ
44
48
エークェント
50
ジャーナル
46
バックアップ・アプライア
56
バックアップ・レシーバ
54
ジャーナル
58
52
レプリカ・データベース
60

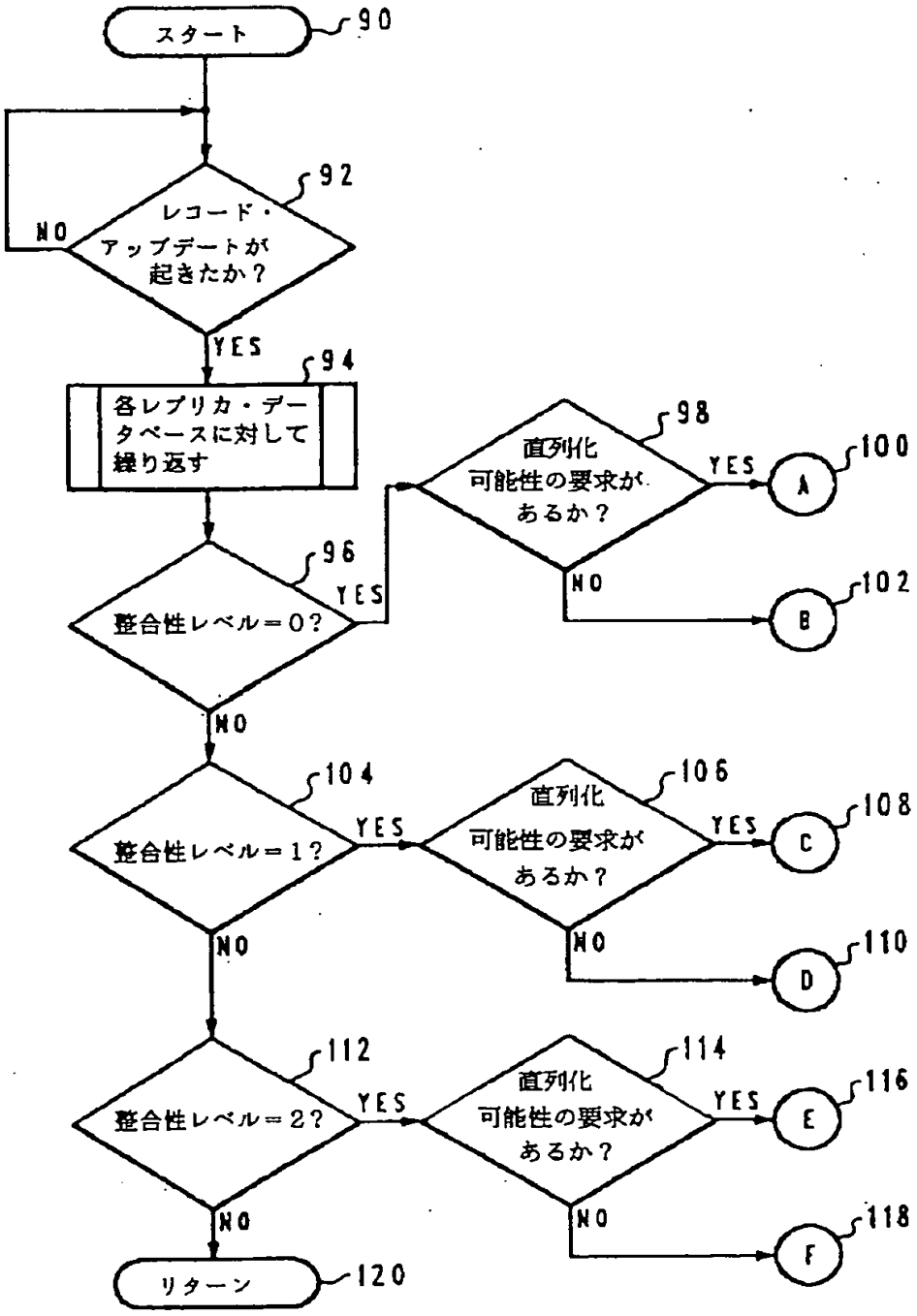
【図５】


A
100
プライマリ・データベース
130
ジャーナル・エントリのフロー
132
バックアップ・レシーバ
134
バックアップ・アプライア
136
トランザクションT1 アップデートU1 レコードR1
150
JE1
アクノレッジメントを送る
JE1をローカルジャーナルに入れる
R1をロックする
JE1を適用する
トランザクションT2 アップデートU2 レコードR2
152
JE2
アクノレッジメントを送る
JE2をローカルジャーナルに入れる
R2をロックする
JE2を適用する
トランザクションT1 アップデートU3 レコードR3
154
JE3
アクノレッジメントを送る
JE3をローカルジャーナルに入れる
R3をロックする
JE3を適用する
トランザクションT1 コミット
156
JE4
アクノレッジメントを送る
JE4をローカルジャーナルに入れる
R1をリリースする
R3をリリースする

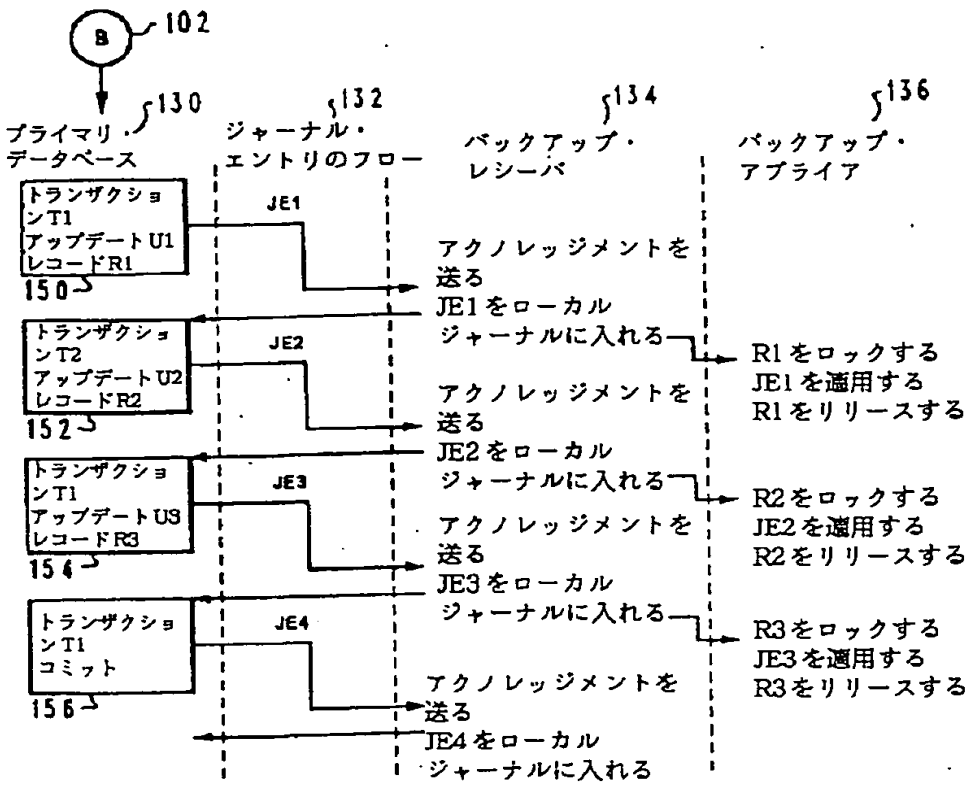
【図３】


スタート
70
整合性
レベルの選択が
行われたか？
72
NO
YES
最初のレプリカ・
データベース
74
所望の整合性レベルの
入力を促す
76
入力されたか？
78
NO
YES
直列化可能性要求の
入力を促す
80
入力されたか？
82
NO
YES
最後の
レプリカ・データ
ベース？
84
NO
YES
リターン
86

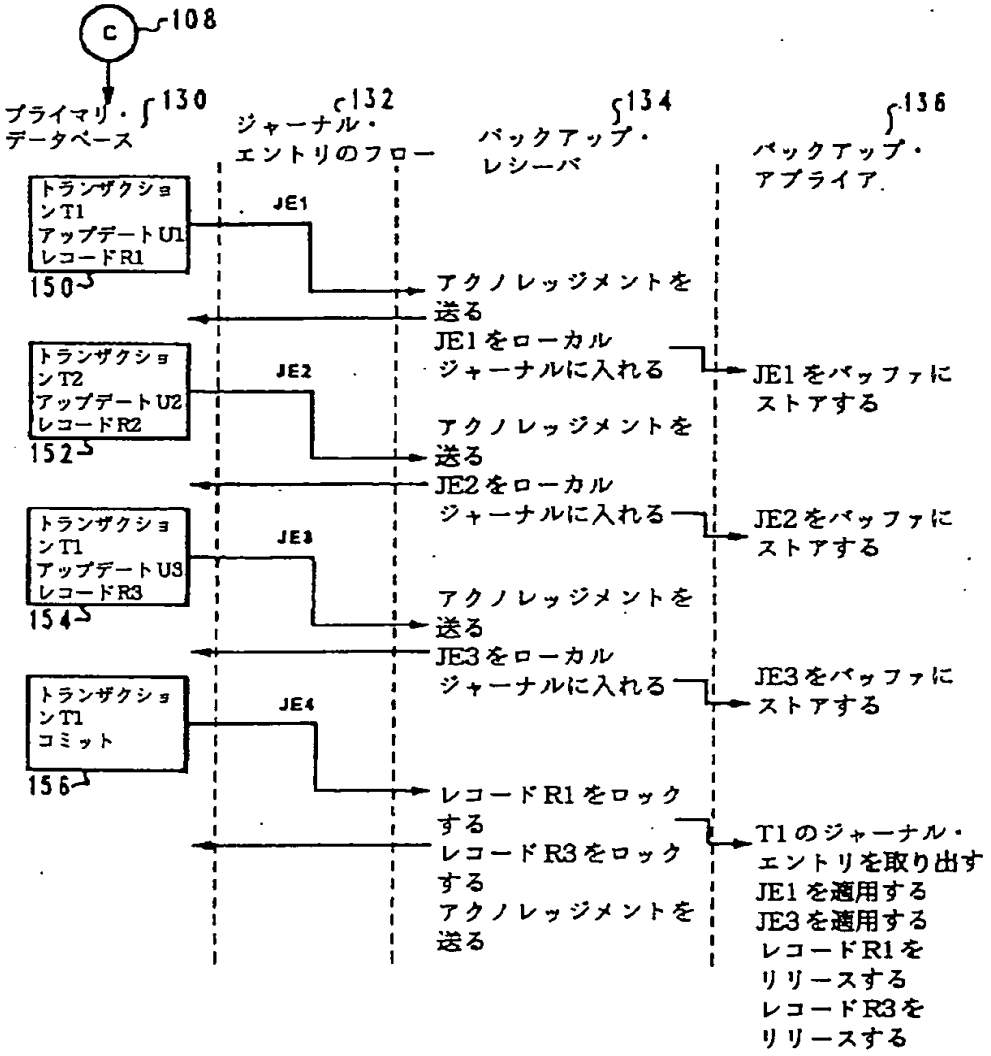
【図４】


スタート
90
レコード・アップデートが起きたか？
92
NO
YES
各レプリカ・データベースに対して繰り返す
94
整合性レベル＝０？
96
YES
NO
直列化可能性の要求があるか？
98
YES
A
100
NO
B
102
整合性レベル＝１？
104
YES
NO
直列化可能性の要求があるか？
106
YES
C
108
NO
D
110
整合性レベル＝２？
112
YES
NO
直列化可能性の要求があるか？
114
YES
E
116
NO
F
118
リターン
120

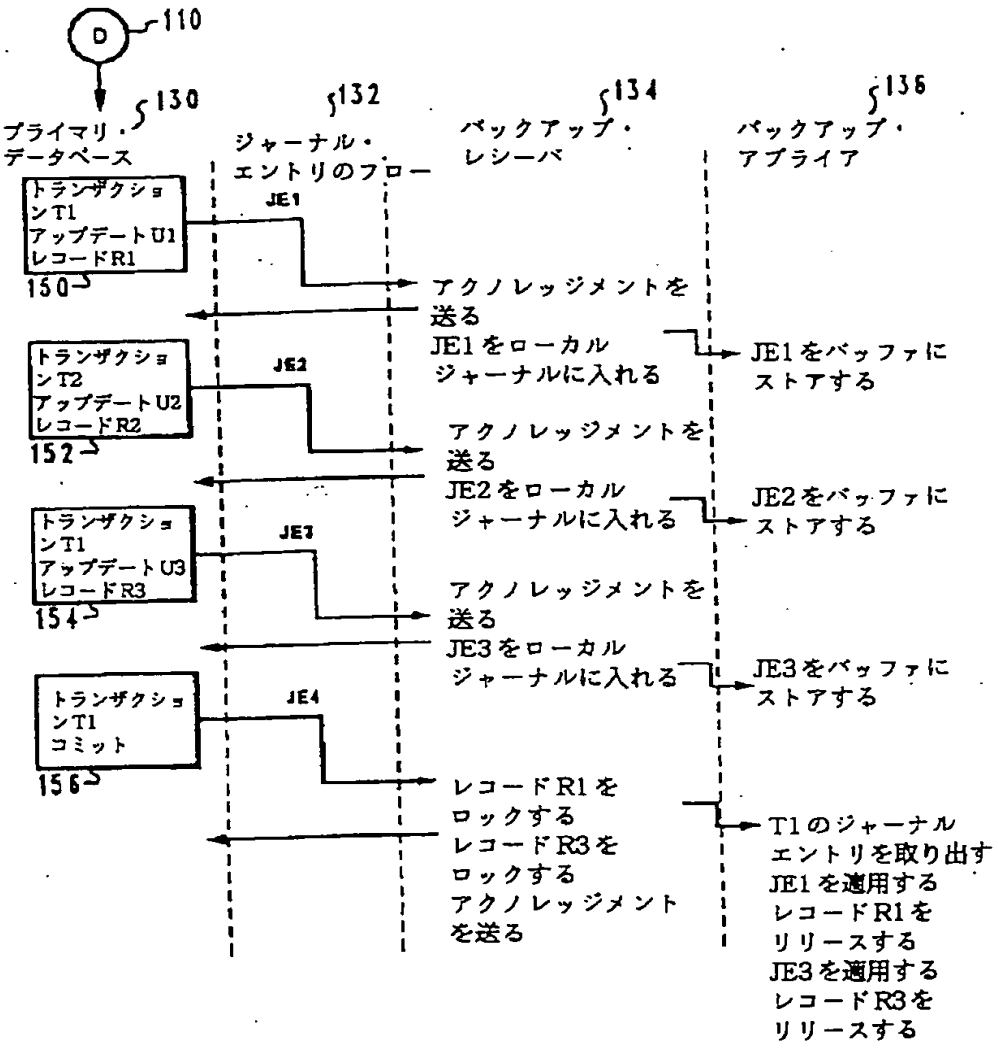
【図６】


B
102
130
プライマリ・データベース
132
ジャーナル・エントリのフロー
134
バックアップ・レシーバ
136
バックアップ・アプライア
トランザクションT1 アップデートU1 レコードR1
150
JE1
アクノレッジメントを送る
JE1をローカルジャーナルに入れる
R1をロックする
JE1を適用する
R1をリリースする
トランザクションT2 アップデートU2 レコードR2
152
JE2
アクノレッジメントを送る
JE2をローカルジャーナルに入れる
R2をロックする
JE2を適用する
R2をリリースする
トランザクションT1 アップデートU3 レコードR3
154
JE3
アクノレッジメントを送る
JE3をローカルジャーナルに入れる
R3をロックする
JE3を適用する
R3をリリースする
トランザクションT1 コミット
156
JE4
アクノレッジメントを送る
JE4をローカルジャーナルに入れる

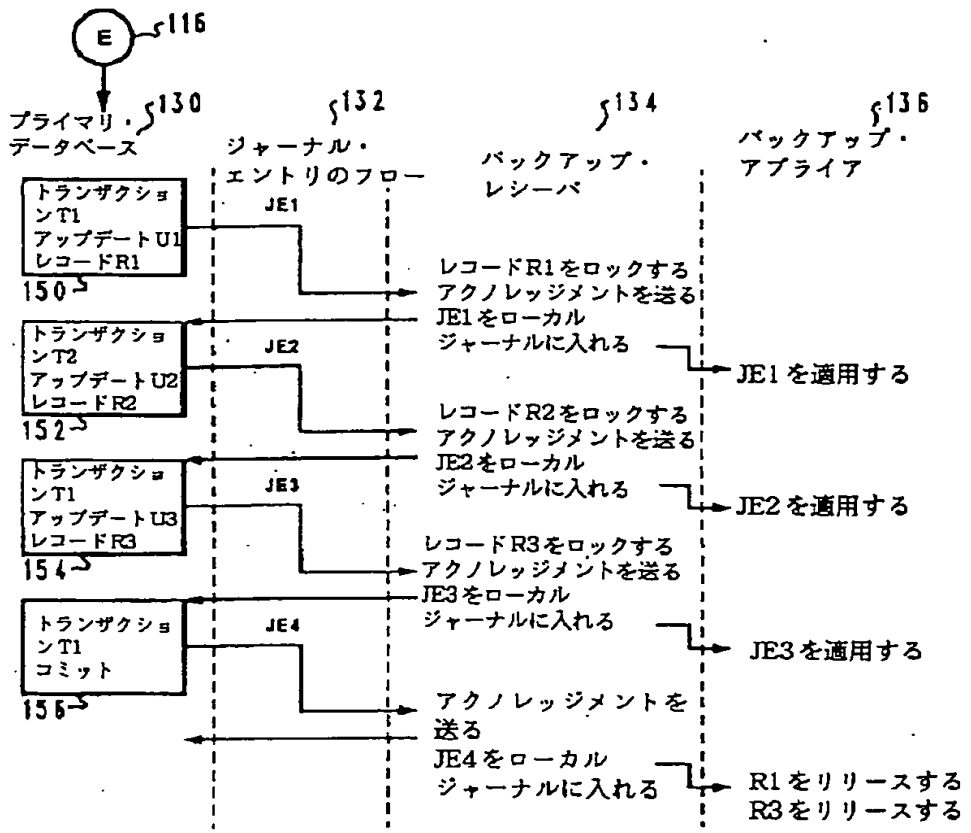
【図７】


C
108
プライマリ・データベース
130
ジャーナル・エントリのフロー
132
バックアップ・レシーバ
134
バックアップ・アプライア
136
トランザクションT1 アップデートU1 レコードR1
150
JE1
アクノレッジメントを送る
JE1をローカルジャーナルに入れる
JE1をバッファにストアする
トランザクションT2 アップデートU2 レコードR2
152
JE2
アクノレッジメントを送る
JE2をローカルジャーナルに入れる
JE2をバッファにストアする
トランザクションT1 アップデートU3 レコードR3
154
JE3
アクノレッジメントを送る
JE3をローカルジャーナルに入れる
JE3をバッファにストアする
トランザクションT1 コミット
156
JE4
レコードR1をロックする
レコードR3をロックする
アクノレッジメントを送る
T1のジャーナル・エントリを取り出す
JE1を適用する
JE3を適用する
レコードR1をリリースする
レコードR3をリリースする

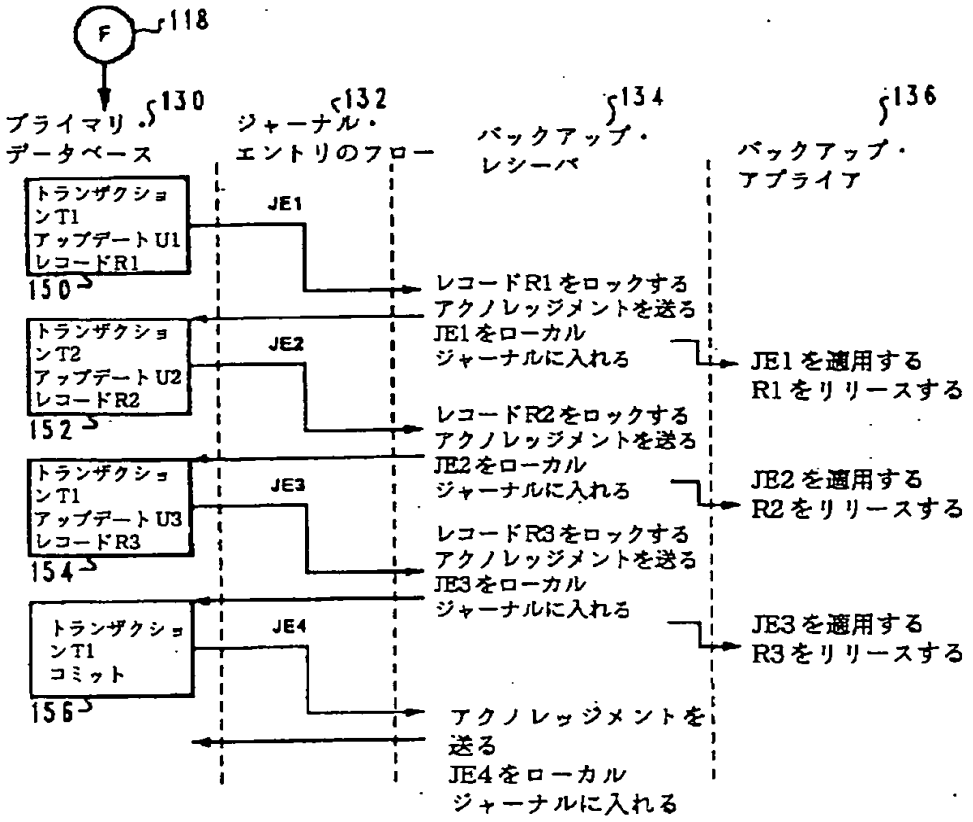
【図8】


D
110
プライマリ・データベース
130
ジャーナル・エントリのフロー
132
バックアップ・レシーバ
134
バックアップ・アプライア
136
トランザクションT1 アップデートU1 レコードR1
150
JE1
アクノレッジメントを送る
JE1をローカルジャーナルに入れる
JE1をバッファにストアする
トランザクションT2 アップデートU2 レコードR2
152
JE2
アクノレッジメントを送る
JE2をローカルジャーナルに入れる
JE2をバッファにストアする
トランザクションT1 アップデートU3 レコードR3
154
JE3
アクノレッジメントを送る
JE3をローカルジャーナルに入れる
JE3をバッファにストアする
トランザクションT1 コミット
156
JE4
レコードR1をロックする
レコードR3をロックする
アクノレッジメントを送る
T1のジャーナルエントリを取り出す
JE1を適用する
レコードR1をリリースする
JE3を適用する
レコードR3をリリースする

【図９】


E
116
プライマリ・データベース
130
ジャーナル・エントリのフロー
132
バックアップ・レシーバ
134
バックアップ・アプライア
136
トランザクションT1 アップデートU1 レコードR1
150
JE1
レコードR1をロックする
アクノレッジメントを送る
JE1をローカルジャーナルに入れる
JE1を適用する
トランザクションT2 アップデートU2 レコードR2
152
JE2
レコードR2をロックする
アクノレッジメントを送る
JE2をローカルジャーナルに入れる
JE2を適用する
トランザクションT1 アップデートU3 レコードR3
154
JE3
レコードR3をロックする
アクノレッジメントを送る
JE3をローカルジャーナルに入れる
JE3を適用する
トランザクションT1 コミット
156
JE4
アクノレッジメントを送る
JE4をローカルジャーナルに入れる
R1をリリースする
R3をリリースする

【図１０】



フロントページの続き

(72)発明者　ダニー・ドレブ
イスラエル、メバセレット・イェルチャライム、メボ・ナアマ　28
(72)発明者　ジャーマン・ゴフト
アメリカ合衆国13760ニューヨーク州エンディコット、ボーント・ヒル・ロード　207　アパートメント　23
(72)発明者　ジョン・エム・マーバーグ
イスラエル34754ハイファ、ビトキン・ストリート　31
(72)発明者　ジェイムス・ジー・ランワイラー
アメリカ合衆国55902ミネソタ州ロチェスター、グレンクロフト・レイン・サウスウェスト　5720
(72)発明者　ジュリアン・サトラン
アメリカ合衆国10589ニューヨーク州ソマーズ、ヘリテッジ・ヒルズ　82エイ